BAB II
LANDASAN TEORI

2.1 **Penelitian Sebelumnya**

Pada bab ini berisi landasan teori sebagai rujukan untuk dilaksanakanna penelitian ini. Rujukan utama penelitian ini adalah penelitian dari Wilma Waterlander yang berjudul *The virtual supermarket: An innovatice research.* Pada penelitian tersebut Wilma Waterlander meneliti penggunaan *virtual reality* pada supermarket. Hasil yang didapatkan adalah penggunaan *virtual reality* mudah dipahami, tetapi masih memiliki kendala dalam melakukan pengaturan data pada *virtual reality* [**2**]. Pada **Tabel 2.1** adalah perbedaan – perbedaan penelitan ini dan penelitian dari Wilma Waterlander yang dijabarkan dalam tabel berikut:

Tabel 2.1 Perbedaan dengan Penelitian Sebelumnya

Implementasi RESTFUL dan Sistem Manajemen pada Virtual Shop Berbasis 3D	**The virtual supermarket: An innovative research**
a. Menggunakan web service berbasis *RESTFUL* untuk menyediakan data. b. Penyimpanan data dilakukan dengan menggunakan *web service* agar data antara satu pengguna dan lain tersinkronasi. c. Memilikik sistem manajemen data.	a. Tidak Menggunakan *web service.* b. Penyimpanan data dilakukan pada sisi pengguna dengan menggunakan *spreadsheet* dengan format *csv.* c. Tidak memiliki sistem manajemen data secara keseluruhan.

Adapun untuk pengkajian materi tersebut adalah kajian tentang bisnis proses yang ada pada hasil penelitian sebelumnya. Karena penelitian ini berfokus kepada perancangan sistem manajemen data yang menggunakan *web service*

berbasisi *RESTFUL* serta sistem manajemen maka akan dilakukan pengkajian materi pada jurnal – jurnal yang lain.

2.2 *Web Service*

2.2.1 Definisi *Web Service*

Web Service adalah sistem dari perangkat lunak yang dibuat untuk mempermudah menukar informasi antar perangkat lunak. Web service menggunakan teknologi web yaitu HTTP untuk menjadi perantara pengiriman atau penerimaan data. Format data yang digunakan web service adalah XML atau JSON karena web service berfokus kepada komunikasi antara perangkat lunak. Web service memiliki antarmuka yang digunakan perangkat lunak lain untuk berkouinikasi dengannya.

2.2.2 *HTTP*

HTTP (Hypertext transfer protocol) adalaha sebuah protocol applikasi untuk sistem informasi terdistribusi, kolaboratif, dan multi media. HTTP merupakan protocol yang biasanya digunakan oleh web service untuk proses komunikasi. HTTP memiliki metode yang digunakan untuk mengindikasikan aksi yang akan dilakukan.

a. *GET* merupakan metode HTTP yang dikhususkan untuk mengambil data.
b. *POST* merupakan metode HTTP yang biasanya digunakan pada saat melakukan submit form untuk membuat data baru
c. *PUT* merupakan metode HTTP yang biasanya digunakan pada saat melakukan submit form untuk mengubah data.
d. *DELETE* merupakan metode HTTP yang biasanya digunakan menghapus.

2.2.3 *JSON*

JSON (Javascript object notation) adalah standar terbuka untuk mengirimkan objek yang terdiri dari atribut dan nilai dengan tipe data array. JSON sering digunakan untuk komunikasi asynchronous antara browser dan server. Contoh JSON bisa dilihat pada **Gambar 2.1.**

```
{
  "glossary": {
    "title": "example glossary",
    "GlossDiv": {
      "title": "S",
      "GlossList": {
        "GlossEntry": {
          "ID": "SGML",
          "SortAs": "SGML",
          "GlossTerm": "Standard Generalized Markup Language",
          "Acronym": "SGML",
          "Abbrev": "ISO 8879:1986",
          "GlossDef": {
            "para": "A meta-markup language, used to create markup
languages such as DocBook.",
            "GlossSeeAlso": ["GML", "XML"]
          },
          "GlossSee": "markup"
        }
      }
    }
  }
}{
```

Gambar 2.1 Contoh *JSON*

2.3 ***REST***

REST (Representational state transfer) adalah salah satu arsitektur web service yang berfungsi untuk menyediakan cara menukar informasi antara perangkat lunak melalui HTTP. *REST* atau *RESTFUL* memungkinkan sistem untuk melakukan proses manipulasi representasi textual dari data yang sebenarnya menggunakan URI yang ditentukan secara stateless. *REST* bisa menggunakan JSON sebagain format data yang menjadi representasi [**3**].

2.3.1 ***Uniform Interface***

Uniform interface atau endpoint menyederhanakan serta membedakan *REST* dari arsitektur yang lain. *Uniform interface* memungkinkan untuk mengembangkan bagian – bagian lain secara terpisah. *Uniform interface* pada *REST* menggunakan *URI*. Contoh dengan menggunakan metode *GET* klien *REST* melakukan *request* ke https://api.example.com/resources/ [**3**].

2.3.2 ***Stateless***

Statelss pada *REST* adalah setiap *request* yang di buat klien independen dan tidak tergantung pada *request* yang lainnya. Karena *REST* bersifat stateless maka tidak boleh ada penggunaan session pada server. Jika ingin melakukan *authorization* biasanya digunakan header *Authorization* karena setiap request harus memiliki informasi yang cukup untuk bisa dimengerti *REST*. [**3**].

2.3.3 *Layered System*

Layered system pada *REST* adalah sistem bisa membuat lapisan – lapisan sebelum request dari klien sampai ketujuan. Dengan *Layered system* pengembang bisa membuat kebijakan keamanan [**3**].

2.4 *Virtual Reality*

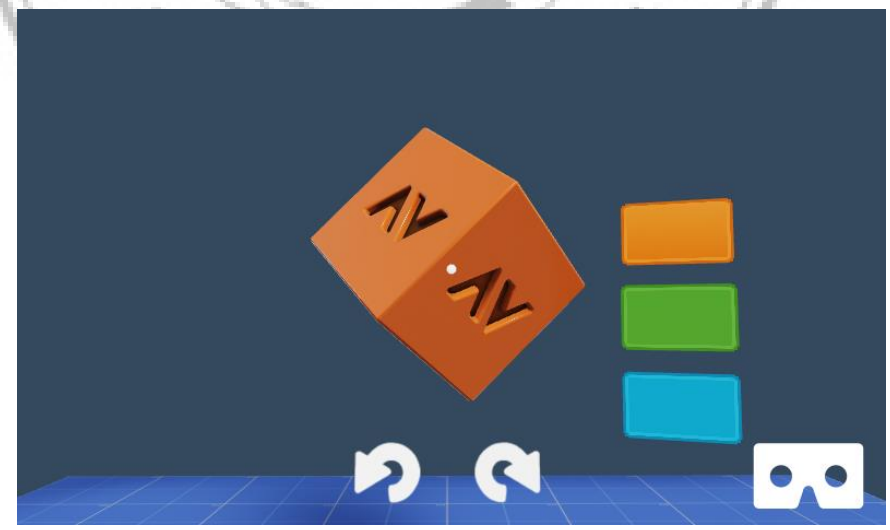
2.4.1 Definisi *Virtual Reality*

Virtual reality adalah sebuah istilah yang digunakan untuk lingkungan 3D dihasilkan oleh computer, yang memungkinkan penggunanya untuk masuk dan berinteraksi dengan realitas alternatif [**4**]. *Virtual reality* mulai digunakan oleh beberapa perusahaan untuk melakukan pemasaran produk mereka contoh *Coca – cola, McDonald, IKEA.* Selain itu perusahaan – perusahaan teknologi besar seperti *Google* dan *Facebook* sedang berlomba untuk mengembangkan *platform* VR merka masing – masing G*oogle* dengan D*aydream* dan F*acebook* dengan *Oculus Rift*.

2.4.2 Komponen *Virtual Reality*

1. Perangkat Input

   Perangkat input adalah alat bagi pengguna VR untuk berinteraksi dengan VR. Sistem akan membaca input dan memberikan reaksi. Perangkat Input ini bisa dikelompokan sebagain alat *point-input, tracking, perangkat-suara, dan bio-controler*. Pada **Gambar 2.2** merupakan contoh point input.



Gambar 2.2 Contoh Point Input

Point-input merupakan input standar yang digunakan dalam VR. Pada **Gambar 2.2** merupakan contoh *point-input*, *point-input* pada cardboard digerakan searah dengan gerakan kepala pengguna. Hal ini mungkin karena ada sensor *gyroskop* dan *akselerometer* pada *smarthphone*. Selain dengan cara yang di atas point-input bisa gunakan dengan *mouse* 3D, atau *joystick*. Tracking merupakan input yang mengirimkan posisi tubuh kita relatif pada saat pertama kita didalam VR [**4**].

2. VR *Engine*

   VR *Engine* adalah pemroses dan penyimpan data. *real-time*, tampilan grafis, dan pemrosesan gambar adalah hal – hal yang perlu diperhitungkan karena pemrosesannya memakan resourse serta waktu. Oleh karena itu harus disesuaikan dengan applikasi VR [**4**].

3. Perangkat Output

   Perangkat Output adalah yang menerima hasil olahan dari VR *Engine* dan memberikan kepengguna melalui alat yang merangsang indra pengguna. Keluaran VR Engine bisa dikelompokan berdasarkan indra yaitu grafis, audio [**4**].

2.5 *Software Framework*

2.5.1 Definisi *Software Framework*

Software Framework adalah platform konseptual yang dimana memiliki kode – kode yang biasanya ada pada setiap tahap pengembangan perangkat lunak. Fungsionalitas umum ini bisa digunakan atau diubah oleh pengembang atau pengguna. Biasanya *framework* ada dalam bentuk beberapa *library* yang sudah dibungkus dengan *API* yang konsisten, serta dapat digunakan dengan baik dalam perangkat lunak yang sedang dikembangkan [**5**]. Beberapa hal yang membuat framework berbeda dengan *library* pada umumnya adalah sebagai berikut.

1. *Default Behavior*: Sebelum dikostumisasi oleh pengembang, *framework* memiliki fungsionalitas yang standar.
2. *Inversion of Control*: Alur kontrol global dalam *framework* ditangani oleh *framewok* bukan oleh pengguna. Hal ini berbeda dengan *library* yang biasanya digunakan saat dipanggil oleh penggunanya.

3. *Extensibility*: Pengguna bisa mengembangkan kode – kode yang ada dalam *framework* dengan cara melakukan *extend* kode asli dalam *framework* dengan kode pengguna.
4. *Non-modifiable Framework Code*: Pengguna bisa melakukan *extend* kode didalam *framework* tetapi tidak dengan mengganti kode *framework* dengan kode pengguna. Hal ini dimaksudkan untuk mempermudah pengguna dalam mengeluarkan usahanya untuk mengembangkan perangkat lunak dan tidak berhadapan dengan hal – hal yang sudah deselesaikan oleh *framework* itu sendiri [**5**].

2.5.2 Keunggulan Menggunakan *Framework*

1. Mengunakan kode – kode yang sudah dibuat sebelumnya dengan kualitas yang bagus. Meningkatkan keandalan dari perangkat lunak serta mempercepat waktu penyelesaian perangkat lunak.
2. Sebuah *Framework* dapat membantu dalam menggunakan *design pettern* dan alat pemrograman.
3. *Framework* menyediakan fungsionalitas baru, meningkatkan performa, meningkatkan kualitas perangkat lunak tanpa pemrograman tambahan oleh pengguna [**5**].

2.5.3 Kekurangan Menggunakan *Framework*

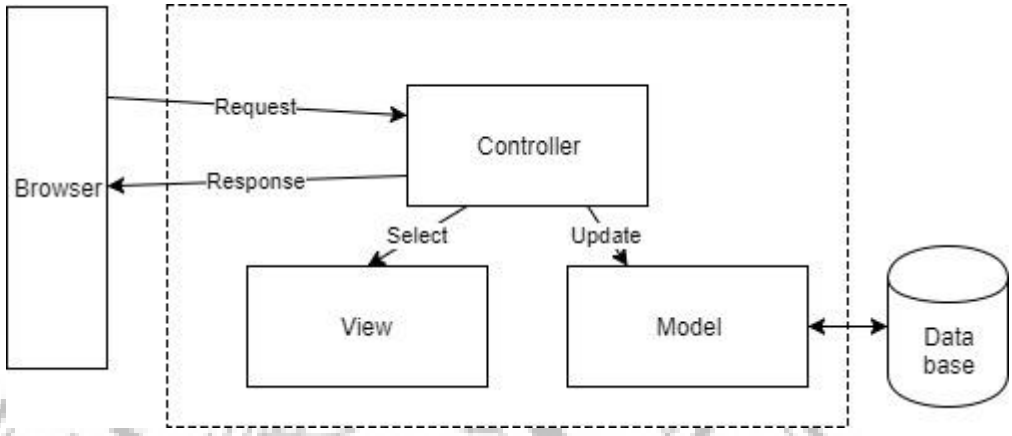
1. Mempelajari satu *framework* bisa memakan waktu dan sulit.
2. Seiring berjalannya waktu kompleksitas perangkat lunak akan semakin tinggi dalam *framework*.
3. *Framework* menambah ukuran perangkat lunak, fenomena ini disebut dengan "*code bloat*" [**5**].

2.5.4 *MVC*

MVC (*Model View Controller*) adalah salah satu jenis arsitektur perangkat lunak. *MVC* membagi perangkat lunak menjadi tiga bagian yang terhubung yaitu *model, view, dan controller*. Setiap bagian pada *MVC* berdiri sendiri dan memiliki fungsi yang jelas. Oleh karena itu *MVC* berguna dalam pengembangan modular serta meningkatkan efisiensi, perawatan sistem dan penggunaan ulang code [**6**].

Berikut pada **Gambar 2.3** digambarkan relasi antara bagian pada *MVC*. Pada saat pertama kali *browser* melakukan *request*, *request* tersebut diterima oleh *controller*. Tergantung dari *route* yang ada pada *request controller* akan

mengakses *model*. *Controller* akan membaca, mengubah, atau menghapus data pada *database* melalui perantara *model*. Setelah itu c*ontroller* akan memilih *view* yang sesuai dengan *request* yang dikirm *browser* dan mengolah *tamplate* pada *view* dan data dari *model* untuk dikirimkan lagi ke browser berupa *response*.



Gambar 2.3 Gambaran umum *MVC*

Model adalah representasi sebuah objek atau pembawa data. *Model* juga bisa memiliki logika untuk mengupdate data dari *database*.

1. View mewakili visualisasi data dari data yang di bawa oleh *model*.
2. Controller mengatur arus data ke objek *model* dan memperbaharui *view* saat data dari *model* berubah. *Controller* membuat *view* dan *model* terpisah [7].

2.6 ***Lumen***

Lumen adalah micro-framework dari laravel yang difokuskan untuk membuat API. Pada Lumen banyak komponen – komponen laravel yang dihilangkan, biasanya komponen – komponen Laravel yang berhubungan dengan View applikasi. Lumen mengklain bahwa memiliki kecepatan yang tinggi dengan bisa menangani 1900 request perdetik.

2.6.1 **Layanan *Lumen***

1. ***Routing***

   Routing merupakan layanan dari *Lumen* yang berfungsi untuk memetakan *URI* dengan *controller* atau *anonymus function*. *Routing* berfungsi untuk mempermudah pengembang dalam melakakun pengaturan pada *endpoint* dari *API* yang nanti ingin dibuat.

2. ***Middleware***

   Middleware dalam Lumen merupkan lapisan – lapisan yang dilewati oleh reques untuk bisa sampai kepada controller. Middleware merupakan mekanisme yang mempermudah pengembang untuk melakukan penyaringan HTTP request yang valid untuk memasuki applikasi nanti.

3. ***Eloquent ORM***

   Eloquent ORM pada Lumen merupakan implementasi dari ActiveRecord berfungsi untuk bekerja dengan database. Eloquent ORM merupakan M dari arsitektur MVC. Setiap Eloquent ORM memiliki model per tabel yang bergunak untuk melakukan query.

2.7 *Json Web Token (JWT)*

JSON Web Token (JWT) adalah representasi dari format klaim yang dimaksudkan untuk ruang lingkungan terbatas seperti header *HTTP* Otorisasi dan parameter permintaan *URI* [**8**].

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

Gambar 2.4 Contoh *JWT*

Pada **Gambar 2.4** bisa dilihat bahwa *JWT* memiliki tiga bagian yang di pisahkan oleh tanda titik (.). Tiga bagian tersebut merupakan hasil dari tiga input yang digunakan untuk membuat signature.

1. *Header*

   Header terdiri dari dua informasi yaitu *JWT* dan algoritma yang akan digunakan untuk melakukan *signature.*

   {

   "alg": "HS256",

   "typ": "JWT"

   }

2. *Payload*

   Payload berisi klaim, klaim adalah informasi dari klien beserta *metadata*.

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

3. *Signature*

   Signature merupakan hasil dari HMACSHA256(base64UrlEncode(header) + "." +base64UrlEncode(payload),secret). Signature berfungsi untuk memastikan bahwa pesan tidak berubah disepanjang jalan.